Flappy Junco:
Real time autonomous flight for a 2-dimensional fixed-wing model

EECE5550 Spring 2023
Matthew Wallace* & Tony Smoragiewicz†
Northeastern University
Department of Electrical and Computer Engineering
360 Huntington Ave., Boston, MA 02115

Project Description

Objective

The following document is a proposal for our final project for EECE5550: Mobile Robotics. We propose to develop a 2-dimensional prototype of the Flight Software (FSW) for the Junco Remote Controlled (RC) aircraft. Junco is an in-development 3.5 Kg RC aircraft intended as a test bed for autonomous operations. Figure 1 shows an early stage render of the vehicle. Junco is build around an Nvidia Jetson Nano, acting as the vehicle's flight computer. The vehicle is designed around a cruise speed of 15 m/s.

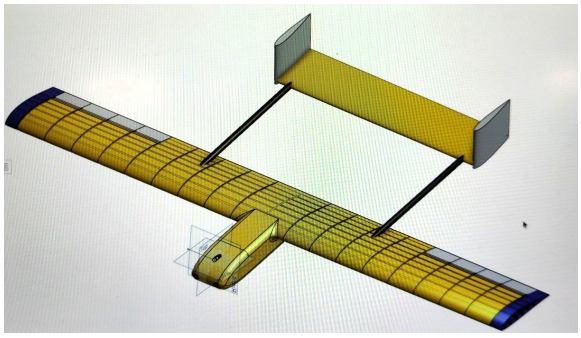

Figure 1: Early render of Junco.

The objective of the project is to prototype the key software subsystems for Junco. Additionally, we aim to test the interaction between high frequency inner loop flight controller, which will stabilize the vehicle with traditional control and navigation approaches, and the lower frequency perception and motion planning system.

Software overview

Figure 2 shows the architecture for the proposed model and simulation. A World Model will simulate the dynamics of the vehicle and track a set of obstacles the vehicle must maneuver around. The left, high frequency portion includes noise generation, which will corrupt state measurements, a navigation block with an Extended Kalman Filter, and an adaptive Linear Quadratic Regulator flight controller. The right, low frequency portion begins with simulating image inputs for the perception algorithm. The perception block will produce an updating occupancy grid, which is used in the path planner to generate an initial path. The CVX TrajOpt block employs a raytracing procedure to determine a feasible region around the path then optimizes a trajectory inside this region, which is handed to the flight controller to track.

Our software will be developed in Python with all subsystems encapsolated in ROS nodes. We plan to demo the simulation on the Jetson Nano. Our architecture is based on that of (Ryll et al. 2019), which is the primary reference paper for the project.

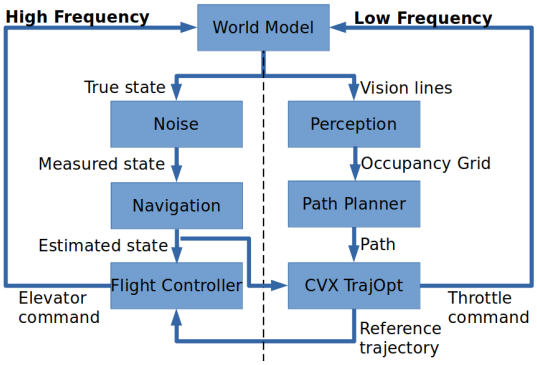

Figure 2: Architecture for the simulation and modeling of the system.

World Model

Dynamics Model

We plan to use a simple 5 state model of the flight dynamics, with states being the flight path angle, the pitch, the airspeed, the horizontal distance, and the vertical height $(\gamma, \theta, V, x, z)$. For a complete treatment of flight dynamics, see (Stevens, Lewis, and Johnson 2015).

The angle of attack is defined as:

$$\alpha = \theta - \gamma$$

*wallace.matth@northeastern.edu
†smoragiewicz.tony@northeastern.edu

The coefficients of lift, drag, and pitching moment are defined as:

$$\begin{aligned} C_L =& C_{L0} + C_{L\alpha}\alpha \\ C_D =& C_{D0} + KC_L^2 \\ C_{\mathcal{M}} =& C_{\mathcal{M}0} + C_{\mathcal{M}\alpha}\alpha + C_{\mathcal{M}\dot{\alpha}}\dot{\alpha} + C_{\mathcal{M}\delta_e}\delta_e \end{aligned}$$

The values for the above constants come from the 2-d wing profile used on Junco. The lift force, drag force, and pitching moment:

$$\begin{aligned} L =& \frac{1}{2}\rho V^2 S C_L \\ D =& \frac{1}{2}\rho V^2 S C_D \\ \mathcal{M} =& \frac{1}{2}\rho V^2 S c C_{\mathcal{M}} \end{aligned}$$

The underlying nonlinear time-invariant dynamical equations used for numerical integration:

$$\begin{aligned} m\dot{V} =& -D - mg\sin\gamma + T\cos\alpha \\ -mV\dot{\gamma} =& -L + mg\cos\gamma + T\sin\alpha \\ \dot{x} =& V\cos\gamma \\ \dot{z} =& -V\sin\gamma \\ \ddot{\theta} I_{yy} =& \mathcal{M} \end{aligned}$$

The vehicle has two control inputs that can be applied at every time step. These are engine thrust (T) and elevator deflection (δ_e). Representation of the control u can be seen below:

$$\begin{aligned} 0 \leq T \leq T_{max} \\ -\delta_{e_{max}} \leq \delta_e \leq \delta_{e_{max}} \end{aligned}$$

Obstacle Model

Obstacles will appear in the simulation as planner convex shapes. The convexity will allow for faster collision checking. Once detected, the objects will be used to fill a occupancy grid map of the environment.

High Frequency Loop

Noise

The noise is a Gaussian disturbance added to measurements of the flight angles and airspeed. We will characterise the variances using test data from Junco's sensors, which include an airspeed indicator, IMU, and pitch gyroscope. The noise block will generate measured airspeed, pitch angle, and position delta to be used in navigation.

Navigation

The navigation block contains an Extended Kalman Filter and estimates the true states based on simulated IMU and airspeed indicator measurements. The estimated state is used both in the flight controller and trajectory generation.

Flight Controller

The systems flight controller is an adaptive LQR. We linearize offline based on flight path angle, pitch angle, and airspeed. The flight controller is responsible for tracking the desired pitch angle determined by the CVX TrajOpt.

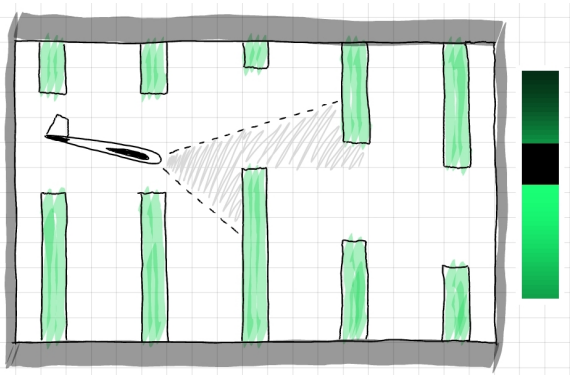

Figure 3: Sample drawing of the simulation. The grey slice represents the field-of-view of the front-facing camera. The vertical bar on the right side is the projection from the UAV's camera view. Note that the green obstacles are spaced further apart in the simulation.

Low Frequency Loop

Perception

The Perception block will take in images from a pair of cameras mounted at a quarter of the span from the fuselage. However, for our simulation, the drone will be using an artificially generated obstacle field. The drone will be able to identify the objects will a particular angle and range error proportional to the range.

Path Planner

The Path Planner block will employ Jump Point Search to find a feasible path through the obstacle grid supplied by perception.

CVX TrajOpt

CVX TrajOpt block is based on a novel procedure to convert a feasible path into a feasible trajectory for the vehicle to track. First ray tracing is performed around the path to determine a feasible region as outlined in (Meerpohl, Rick, and Büskens 2019). The resulting QP is easy to solve efficiently with solvers such as CVX. The resulting throttle commands are used open-loop while the desired pitch angle is tracked by the flight controller.

References

Meerpohl, C.; Rick, M.; and Büskens, C. 2019. Free-space Polygon Creation based on Occupancy Grid Maps for Trajectory Optimization Methods. *IFAC-PapersOnLine*, 52(8): 368–374. 10th IFAC Symposium on Intelligent Autonomous Vehicles IAV 2019.

Ryll, M.; Ware, J.; Carter, J.; and Roy, N. 2019. Efficient Trajectory Planning for High Speed Flight in Unknown Environments. In *2019 International Conference on Robotics and Automation (ICRA)*, 732–738.

Stevens, B. L.; Lewis, F. L.; and Johnson, E. N. 2015. *Aircraft control and simulation: dynamics, controls design, and autonomous systems*. John Wiley & Sons.

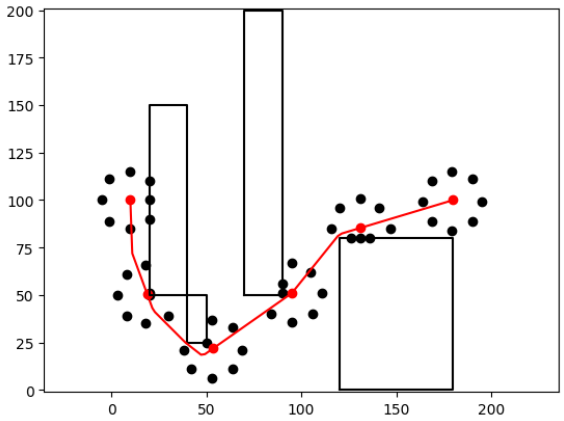

Figure 4: The initial path is converted to a trajectory through the solution of a convex optimization problem. This problem is composed of the standard QP for solving system dynamics linearized around points on the path with linear constraints giving a safe region. The ray tracing proceedure to find this safe region is shown above.